

4.

[English](#) | 

harness 



```
import { Service, type Context } from '@deepseek-ai/cordis'

declare module '@deepseek-ai/cordis' {
  interface Context {
    stats: StatsService
  }
  interface Events {
    'stats/report'(name: string, count: number): void
  }
}

export class StatsService extends Service {
  private counts = new Map<string, number>()

  constructor(ctx: Context) {
    super(ctx, 'stats')
  }

  bump(name: string) {
    const next = (this.counts.get(name) ?? 0) + 1
    this.counts.set(name, next)
    this.ctx.emit('stats/report', name, next)
  }
}
```

```
export const name = 'stats'

export function apply(ctx: Context) {
  ctx.plugin(StatsService)
}
```

interface Events interface Context ctx.emit ctx.on namespace/action

reporter.ts

```
import type { Context } from '@deepseek-ai/cordis'
import type {} from './stats.ts'

export const name = 'reporter'
export const inject = ['stats']

export function apply(ctx: Context) {
  ctx.on('stats/report', (name, count) => {
    console.log(`[stats] ${name} -> ${count}`)
  })
  ctx.stats.bump('tool_call')
  ctx.stats.bump('tool_call')
  ctx.stats.bump('prompt')
}
```

import type {} from './stats.ts' TypeScript

- name: './stats.ts'
- name: './reporter.ts'

[stats] tool_call -> 1
[stats] tool_call -> 2
[stats] prompt -> 1

ctx.on() effect removeListener



emit 5

API	Signature	Behavior
emit	ctx.emit(name, ...args)	promise
parallel	await ctx.parallel(name, ...args)	
serial	await ctx.serial(name, ...args)	null / false / undefined
bail	ctx.bail(name, ...args)	serial
waterfall	ctx.waterfall(name, ...args, next)	

harness

waterfall

waterfall continuation next() waterfall-demo.ts

```
import type { Context } from '@deepseek-ai/cordis'

declare module '@deepseek-ai/cordis' {
  interface Events {
    'demo/transform'(input: string, next: () => Promise<string>): Promise<string>
  }
}

export const name = 'waterfall-demo'

export function apply(ctx: Context) {
  // Listener 1: wrap the downstream result.
  ctx.on('demo/transform', async (input, next) => {
    const downstream = await next()
    return downstream.toUpperCase()
  })

  // Listener 2: short-circuit when it owns the decision.
  ctx.on('demo/transform', async (input, next) => {
    if (input.includes('blocked')) return '** blocked **'
    return next()
  })

  void (async () => {
```

```
console.log(await ctx.waterfall('demo/transform', 'hello', async () => 'hello'))
console.log(await ctx.waterfall('demo/transform', 'blocked words', async () => 'blocked words'))
})()
}
```

cordis.yml

HELLO

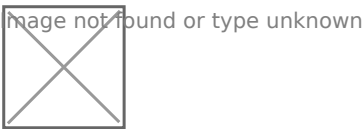
** BLOCKED **

next() blocked next() ctx.waterfall 1

we next() next() waterfall

harness waterfall agent/request approval/request

cordis.yml



#1
3 2026 16:58:44
3 2026 17:02:14